1. A method of allocating registers in an assembler, comprising:

processing assembler code to avoid a register bank allocation error including at least one of a register bank conflict and an insufficient number of physical registers in target hardware; and

automatically manipulating instructions to avoid the register bank allocation error.

2. The method according to claim 1, wherein the register bank conflict is associated with instructions in which first and second operands have respective first and second source registers located in a first one of first and second register banks and further including inserting an instruction to assign the first operand to a temporary register.

3. The method according to claim 2, wherein the register bank conflict is associated with instructions in which first and second operands have respective first and second source registers located in a first one of first and second register banks and further including inserting an instruction to move the first source register to local memory.

4. The method according to claim 1, further including coloring a register graph to detect the register bank conflict.

5. The method according to claim 4, further including identifying registers adjacent to each other in the graph having the same color.

6. The method according to claim 5, further including finding the shortest path having an odd length connecting the registers adjacent to each other having the same color.

7. The method according to claim 6, further including sorting a list of edges in the graph associated with path.

8. The method according to claim 7, further including sorting the list based upon a weight of the edges.

9. The method according to claim 8, further including repeating the finding and sorting to find further solutions to color the graph.

10. The method according to claim 1, further including manipulating instructions to spill one or more registers associated with the assembler code to alternative memory in the target hardware.

11. The method according to claim 10, further including mapping virtual registers to physical registers and spilling a sufficient number of physical registers to enable mapping between the virtual registers and the physical registers in the target hardware.

12. The method according to claim 10, wherein the non-register memory includes one or more of local memory, SRAM memory and DRAM memory.

13. The method according to claim 10, further including identifying registers that should not be spilled.

14. The method according to claim 10, further including determining first and second banks of abstract physical registers for target hardware having alternative memory with a single read port.

15. The method according to claim 14, further including assigning weights to entries in the array corresponding to a number of instructions that reference a physical register.

16. The method according to claim 15, further including sorting the array entries based upon the assigned weights.

17. An article, comprising:

a storage medium having stored thereon instructions that when executed by a machine result in the following:

processing assembler code to avoid a register bank allocation error including at least one of a register bank conflict and insufficient number of physical registers in target hardware; and

automatically manipulating instructions to avoid the register bank allocation error.

18. The article according to claim 17, wherein the register bank conflict is generated by instructions in which first and second operands have respective first and second source registers located in a first one of first and second register banks and further including inserting instructions to assign the first operand to a temporary register and/or local memory.

19. The article according to claim 18, further including stored instructions to color a register graph to detect the register bank conflict.

20. The article according to claim 19, further including stored instructions to spill one or more virtual registers associated with the assembler code.

21. The article according to claim 20, further including stored instruction to spill a sufficient number of registers so that non-spilled ones of the registers can be mapped to physical registers in the target hardware.

22. The article according to claim 21, further including stored instructions to identify registers that should not be spilled.

23. A development/debugger system, comprising:

an assembler to generate microcode that is executable in a processing element by processing assembler code to avoid a register bank allocation error including at least one of a register bank conflict and insufficient number of physical registers in target hardware; and

automatically manipulating instructions to avoid the register bank allocation error.

24. The system according to claim 23, wherein the register bank conflict is generated by instructions in which first and second operands have respective first and second source registers located in a first one of first and second register banks and further including inserting instructions to assign the first operand to a temporary register and/or local memory.

25. The system according to claim 24, wherein the register bank conflict is generated by the insufficient number of physical registers and wherein manipulating the instructions includes spilling one or more of the physical registers to alternative memory.

26. A network forwarding device, comprising:

at least one line card to forward data to ports of a switching fabric;

the at least one line card including a network processor having multi-threaded microengines configured to execute microcode, wherein the microcode comprises a microcode developed using an assembler that

processed assembler code to avoid a register bank allocation error including at least one of a register bank conflict and insufficient number of physical registers in target hardware; and

automatically manipulated instructions to avoid the register bank allocation error.

27. The device according to claim 26, wherein the register bank conflict was generated by instructions in which first and second operands have respective first and second source registers located in a first one of first and second register banks and further including inserting an instruction to assign the first operand to a temporary register and/or local memory.

28. The device according to claim 27, wherein the register bank conflict was generated by the insufficient number of physical registers and wherein the inserted instructions include spilling one or more of the physical registers to alternative memory.

5